RESEARCH WORK ON THE BELOW LISTED TERMINOLOGIES IN RELATIONS TO DATABASE.

1. Cardinality
2. Entity, entity type and entity set
3. Schema
4. DBMS.
5. RDBMS.
6. Normalization
7. DDL, DML, DCL
8. Field, record, tuple and table
9. E-R model
10. Attributes and relations
11. Database transaction

BY

JUSTICE ISREAL AGBONMA

DATABASE MANAGEMENT SYSTEM (DBMS)

As the name suggests, the database management system consists of two parts. They are:

1. Database and

2. Management System

What is a Database?

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

Data: Facts, figures, statistics etc. having no particular meaning

(e.g. 1, ABC, 19 etc).

Record: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll	Name	Age
1	ABC	19

Table or Relation: Collection of related records.

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

The columns of this relation are called Fields, Attributes or Domains. The rows are called Tuples or Records.

Database: Collection of related relations. Consider the following collection of tables:

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Name
1	ABC
2	DEF
3	XYZ

T3

Roll	Year
1	2
2	5
3	6

We now have a collection of 3 tables. They can be called a “related collection” because we can clearly find out that there are some common attributes existing in a selected pair of tables. Because of these common attributes we may combine the data of two or more tables together to find out the complete details of a student. Questions like “Which hostel does the youngest student live in?” can be answered now, although Age and Hostel attributes are in different tables.

A database in a DBMS could be viewed by lots of different people with different responsibilities. For example, within a company there are different departments, as well as customers, who each need to see different kinds of data. Each employee in the company will have different levels of access to the database with their own customized front-end application. In a database, data is organized strictly in row and column format. The rows are called Tuple or Record. The data items within one row may belong to different data types. On the other hand, the columns are often called Domain or Attribute. All the data items within a single attribute are of the same data type.

Management System

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. This is a collection of related data with an implicit meaning and hence is a database. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have implicit meaning. The management system is important because without the existence of some kind of rules and regulations it is not possible to maintain the database. We have to select the particular attributes which should be included in a particular table; the common attributes to create relationship between two tables; if a new record has to be inserted or deleted then which tables should have to be handled etc. These issues must be resolved by having some kind of rules to follow in order to maintain the integrity of the database. Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Database Management System (DBMS) and Its Applications:

A Database management system is a computerized record-keeping system. It is a repository or a container for collection of computerized data files. The overall purpose of DBMS is to allow the users to define, store, retrieve and update the information contained in the database on demand. Information can be anything that is of significance to an individual or organization. Databases touch all aspects of our lives. Some of the major areas of application are as follows:

1. Banking 2.
2. Airlines 3.
3. Universities
4. 4. Manufacturing and selling
5. 5. Human resources

Enterprise Information

- Sales: For customer, product, and purchase information.
- Accounting: For payments, receipts, account balances, assets and other accounting information.
- Human resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for

items. Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Banking and Finance

- Banking: For customer information, accounts, loans, and banking transactions.
- Credit card transactions: For purchases on credit cards and generation of monthly statements.
- Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- Universities: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Purpose of Database Systems

Database systems arose in response to early methods of computerized management of commercial data. As an example of such methods, typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

- Add new students, instructors, and courses
- Register students for courses and generate class rosters
- Assign grades to students, compute grade point averages (GPA), and generate transcripts

System programmers wrote these application programs to meet the needs of the university. New application programs are added to the system as the need arises.

For example, suppose that a university decides to create a new major (say, computer science).As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems. Keeping organizational information in a file processing system has a number of major disadvantages:

- **Data redundancy and inconsistency.** Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages.

Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

- **Difficulty in accessing data:** Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of all students. The university clerk has now two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory. The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

- **Data isolation:** Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Integrity problems:** The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.
- **Atomicity problems:** A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department A but was not credited to the balance of department B, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be atomic—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system. Concurrent-access anomalies. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers. In such

an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider department A, with an account balance of $10,000. If two department clerks debit the account balance (by say $500 and $100, respectively) of department A at almost exactly the same time, the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively. Depending on which one writes the value last, the account balance of department A may contain either $9500 or $9900, rather than the correct value of $9400. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

- **Security problems:** Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult. These difficulties, among others, prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems.

Advantages of DBMS:

- **Controlling of Redundancy:** Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.
- **Improved Data Sharing:** DBMS allows a user to share the data in any number of application programs.
- **Data Integrity:** Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database. For example: in customer database we can can enforce an integrity that it must accept the customer only from Noida and Meerut city.
- **Security:** Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.
- **Data Consistency:** By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.
- **Efficient Data Access :** In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing

- **Enforcements of Standards:** With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.
- **Data Independence:** Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the meta data obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.
- **Reduced Application Development and Maintenance Time:** DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

Disadvantages of DBMS

1) It is bit complex. Since it supports multiple functionality to give the user the best, the underlying software has become complex. The designers and developers should have thorough knowledge about the software to get the most out of it.
2) Because of its complexity and functionality, it uses large amount of memory. It also needs large memory to run efficiently.
3) DBMS system works on the centralized system, i.e.; all the users from all over the world access this database. Hence any failure of the DBMS, will impact all the users.
4) DBMS is generalized software, i.e.; it is written work on the entire systems rather specific one. Hence some of the application will run slow.

DATABASE SCHEMA

A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities. Schemas commonly use visual representations to communicate the architecture of the database, becoming the foundation for an organization’s data management discipline. This process of database schema design is also known as data modeling.

These data models serve a variety of roles, such as database users, database administrators, and programmers. For example, it can help database administrators manage normalization processes to avoid data duplication. Alternatively, it can enable analysts to navigate these data structures to conduct reporting or other valuable business analyses. These diagrams act as valuable documentation within the database management system (DBMS), ensuring alignment across various stakeholders.

Database schema vs. database instance

A database schema is considered the “blueprint” of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data.

A sample of data from a database at a single moment in time is known as a database instance. It contains all the properties that the schema describes as data values. Since database instances are just a snapshot at a given moment, they're likely to change over time, unlike database schemas.

Types of database schemas

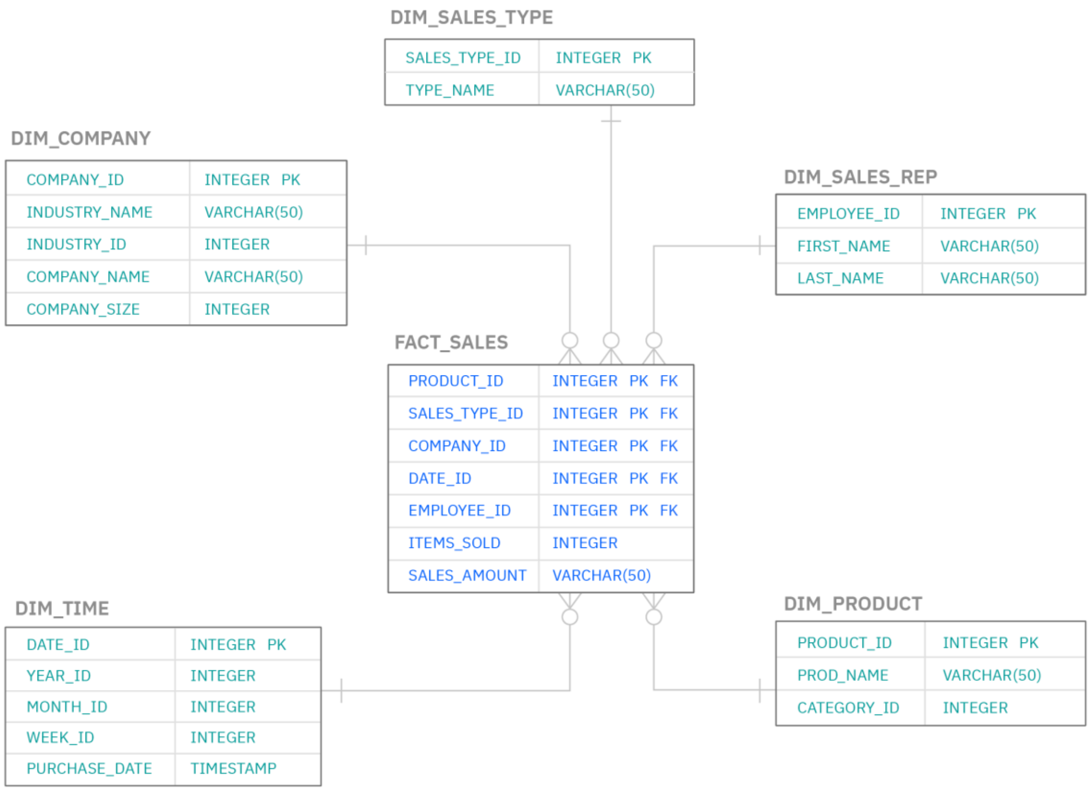
While the term schema is broadly used, it is commonly referring to three different schema types, conceptual database schema, logical database schema, and physical database schema.

- Conceptual schemas offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements.
- Logical database schemas are less abstract, compared to conceptual schemas. They clearly define schema objects with information, such as table names, field names, entity relationships, and integrity constraints—i.e. any rules that govern the database. However, they do not typically include any technical requirements.
- Physical database schemas provide the technical information that the logical database schema type lacks in addition to the contextual information, such as table names, field names, entity relationships, et cetera. That is, it also includes the syntax that will be used to create these data structures within disk storage.

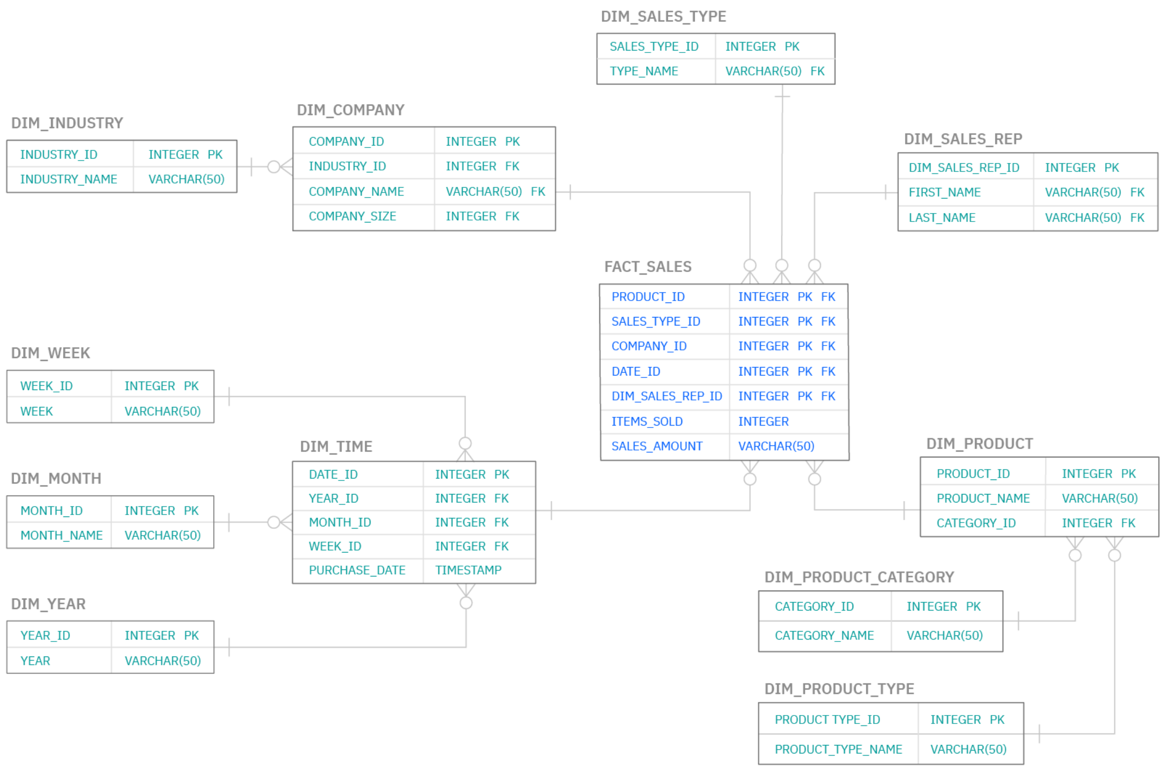
Star schema vs. snowflake schema

In both logical schemas and physical schemas, database tables will have a primary key or a foreign key, which will act as unique identifiers for individual entries in a table. These keys are used in SQL statements to join tables together, creating a unified view of information. Schema diagrams are particularly helpful in showing these relationships between tables, and they enable analysts to understand the keys that they should join on. There are two additional types of schemas are also commonly referenced in the context of relational database management systems (RDBMS); these are known as star schemas and snowflake schemas.

While conceptual, logical, and physical schemas contain different levels of information about databases in their diagrams, star and snowflake schemas represent those relationships between entities differently. More specifically, a star schema is a type of relational database schema that is composed of a single, central fact table that is surrounded by dimension tables. This tends to e considered a simpler schema compared to the snowflake schema.



A snowflake schema consists of one fact table that is connected to many dimension tables, which can be connected to other dimension tables through a many-to-one relationship. This schema offers the advantage of low levels of data redundancy but is not as effective when it comes to query performance.



As the name implies, a star schema tends to look like a star whereas a snowflake schema tends to resemble a snowflake.

Benefits of database schemas

As big data continues to grow, database objects and schemas are critical to ensure efficiency in day-to-day company operations. If relational models are poorly organized and poorly documented, they will be harder to maintain, posing problems for both its users and the company.

Some key benefits of database schemas include:

- **Access and security:** Database schema design helps organize data into separate entities, making it easier to share a single schema within another database. Administrators can also control access through database permissions, adding another layer of security for more proprietary data. For example, a single schema may contain personally identifiable information (PII), which you would want to encrypt for privacy and security purposes.
- **Organization and communication:** Documentation of database schemas allow for more organization and better communication among internal stakeholders. Since it provides a common source of truth, it enables users to understand the logical constraints and methods of aggregation across tables.
- **Integrity:** This organization and communication also helps to ensure data validity. For example, it can help administrators manage normalization processes to avoid data duplication. It can also assist in monitoring compliance of the constraints in the schema's database design, enabling adherence to ACID properties (atomicity, consistency, isolation, durability).

DATA MODELS

Underlying the structure of a database is the data model: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical, and view levels. The data models can be classified into four different categories:

1) **Relational Model:** The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.
2) **Entity-Relationship Model:** The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The entity relationship model is widely used in database design.

3) **Object-Based Data Model:** Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

4) **Semi-structured Data Model:** The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semi-structured data. Historically, the network data model and the hierarchical data model preceded the relational data model. These models were tied closely to the underlying implementation, and complicated the task of modeling data. As a result they are used little now, except in old database code that is still in service in some places.

Conceptual Database Design - Entity Relationship(ER) Modeling:

Database Design Techniques

1. ER Modeling (Top down Approach)

2. Normalization (Bottom Up approach)

ER Modeling

A graphical technique for understanding and organizing the data independent of the actual database implementation We need to be familiar with the following terms to go further.

Entity

Any thing that has an independent existence and about which we collect data. It is also known as entity type. In ER modeling, notation for entity is given below.

Entity instance

Entity instance is a particular member of the entity type.

Example for entity instance : A particular employee

Regular Entity

An entity which has its own key attribute is a regular entity.

Example for regular entity : Employee.

Weak entity

An entity which depends on other entity for its existence and doesn't have any key attribute of its own is a weak 19 entity.

Example for a weak entity : In a parent/child relationship, a parent is considered as a strong entity and the child is a weak entity. In ER modeling, notation for weak entity is given below.

Attributes

Properties/characteristics which describe entities are called attributes. In ER modeling, notation for attribute is given below.

Domain of Attributes

The set of possible values that an attribute can take is called the domain of the attribute. For example, the attribute day may take any value from the set {Monday, Tuesday ... Friday}. Hence this set can be termed as the domain of the attribute day.

Key attribute

The attribute (or combination of attributes) which is unique for every entity instance is called key attribute. E.g the employee_id of an employee, pan_card_number of a person etc.If the key attribute consists of two or more attributes in combination, it is called a composite key. In ER modeling, notation for key attribute is given below.

Simple attribute

If an attribute cannot be divided into simpler components, it is a simple attribute. Example for simple attribute : employee_id of an employee.

Composite attribute

If an attribute can be split into components, it is called a composite attribute. Example for composite attribute : Name of the employee which can be split into First_name, Middle_name, and Last_name.

Single valued Attributes

If an attribute can take only a single value for each entity instance, it is a single valued attribute. example for single valued attribute : age of a student. It can take only one value for a particular student.

Degree of a Relationship

Degree of a relationship is the number of entity types involved. The n-ary relationship is the general form for degree n. Special cases are unary, binary, and ternary ,where the degree is 1, 2, and 3, respectively. Example for unary relationship : An employee ia a manager of another employee Example for binary relationship : An employee works-for department. Example for ternary relationship : customer purchase item from a shop keeper

Cardinality of a Relationship

Relationship cardinalities specify how many of each entity type is allowed. Relationships can have four possible connectivity as given below.

1. One to one (1:1) relationship

2. One to many (1:N) relationship

3. Many to one (M:1) relationship

4. Many to many (M:N) relationship

The minimum and maximum values of this connectivity is called the cardinality of the relationship

Example for Cardinality –One-to-One (1:1)

Employee is assigned with a parking space. One employee is assigned with only one parking space and one parking space is assigned to only one employee. Hence it is a 1:1 relationship and cardinality is One-To-One (1:1)

Example for Cardinality – One-to-Many (1:N)

Organization has employees One organization can have many employees , but one employee works in only one organization. Hence it is a 1:N relationship and cardinality is One-To-Many (1:N) In ER modeling

Example for Cardinality – Many-to-One (M :1)

It is the reverse of the One to Many relationship. employee works in organization One employee works in only one organization But one organization can have many employees. Hence it is a M:1 relationship and cardinality is Many-to-One (M :1) 23 In ER modeling,

Cardinality – Many-to-Many (M:N)

Students enrolls for courses One student can enroll for many courses and one course can be enrolled by many students. Hence it is a M:N relationship and cardinality is Many-to-Many (M:N)

Relationship Participation

1. **Total**

In total participation, every entity instance will be connected through the relationship to another instance of the other participating entity types

2. **Partial**

Example for relationship participation

Consider the relationship - Employee is head of the department

Advantages and Disadvantages of ER Modeling (Merits and Demerits of ER Modeling)

Advantages

1. ER Modeling is simple and easily understandable. It is represented in business users language and it can be understood by non-technical specialist.

2. Intuitive and helps in Physical Database creation.

3. Can be generalized and specialized based on needs.

4. Can help in database design.

5. Gives a higher level description of the system.

Disadvantages

1. Physical design derived from E-R Model may have some amount of ambiguities or inconsistency. 2. Sometime diagrams may lead to misinterpretations

Database Languages

A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database queries and updates. In practice, the data definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

SQL

SQL stands for Structured Query Language. It is used for storing and managing data in Relational Database Management System (RDBMS). It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables. All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language. SQL allows users to query the database in a number of ways, using English-like statements.

SQL follows the following rules:

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

SQL Process

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how-to interpret the task.
- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.

Advantages of SQL

- High speed
- No coding needed
- Well defined standards

- Portability
- Interactive language
- Multiple data view

SQL Datatype

SQL Datatype is used to define the values that a column can contain. Every column is required to have a name and datatype in the database table.

- Binary datatype
- Numeric datatype
- Extract numeric datatype
- String datatype
- Date datatype

SQL Commands

SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data. SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, and set permission for users.

SQL Commands

DDL	**DML**	**DCL**	**TCL**	**DQL**
Create	Insert	Grant	Commit	Select
Drop	Update	Revoke	Rollback	
After	Delete		Save point	

Truncate				

DATA DEFINITION LANGUAGE (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc. All the command of DDL are auto-committed that means it permanently save all the changes in the database. Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

Data Definition Language (DDL)- CREATE

CREATE It is used to create a new table in the database.

Syntax:

REATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES [...]);

Example:

CREATE TABLE EMPLOYEE (Name VARCHAR2 (20), Email VARCHAR2 (100), DOB DATE);

Data Definition Language (DDL)- Drop

Drop: It is used to delete both the structure and record stored in the table.

Syntax:

DROP TABLE;

Example:

DROP TABLE EMPLOYEE;

Data Definition Language (DDL) ALTER

ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

ALTER TABLE table name ADD column name COLUMN-definition;

ALTER TABLE MODIFY (COLUMN DEFINITION....);

Example:

ALTER TABLE STU_DETAILS ADD (ADDRESS VARCHAR2 (20));

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2 (20));

Data Definition Language (DDL)- TRUNCATE

TRUNCATE: It is used to delete all the rows from the table and freethe space containing the table.

Syntax:

TRUNCATE TABLE table_name;

Example:

TRUNCATE TABLE EMPLOYEE;

DATA-MANIPULATION LANGUAGE

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

There are basically two types:

- Procedural DMLs require a user to specify what data are needed and how to get those data.
- Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.

DML commands are used to modify the database. It is responsible for all form of CHANGES in the database. The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

INSERT

UPDATE

DELETE

Data Manipulation Language – INSERT

INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME (col1, col2, col3,.... col N) VALUES (value1, value2, value3, ... valueN);

OR

INSERT INTO TABLE_NAME VALUES (value1, value2, value3, ... valueN);

Example:

INSERT INTO XYZ (Author, Subject) VALUES ("Sonoo", "DBMS");

Data Manipulation Language – UPDATE

Update: This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

Example:

UPDATE students

SET User_Name = 'Sonoo'

WHERE Student_Id = '3'

Data Control Language

DCL commands are used to GRANT and TAKE BACK authority from any database user.

Here are some commands that come under DCL:

- Grant

- Revoke

Data Control Language – Grant

GRANT: It is used to give user access privileges to a database.

Example:

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

REVOKE: It is used to take back permissions from the user.

Example:

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only. These operations are automatically committed in the database that's why they cannot be used while creatingtables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

Transaction Control Language – COMMIT

Commit: Commit command is used to save all the transactions tothe database.

Syntex:

COMMIT;

Example: DELETE FROM CUSTOMERS WHERE AGE = 25; COMMIT;

Transaction Control Language – Rollback

Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntex:

ROLLBACK;

Example:

DELETE FROM CUSTOMERS WHERE AGE = 25; ROLLBACK;

SAVEPOINT: It is used to roll thetransaction back to a certain point without rolling back the entire transaction.

Syntex:

SAVEPOINT SAVEPOINT_NAME;

Data Query Language

DQL is used to fetch the data from the database. It uses only one command:

SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

   Syntax:

   SELECT expressions FROM TABLES WHERE conditions;

   Example:

   SELECT emp_name FROM employee WHERE age > 20;